Kate Pregler
CS 230
20 May 2018

Milestone Report: Harmonic Combiner

Introduction and Related Work

My goal with the harmonic combiner was to transfer musical style much in the same way that we transferred visual style in the neural style transfer project, and as described in the paper by Gatys et al, "A Neural Algorithm of Artistic Style." While there isn't a direct monetizable application for this that I can think of, it might help us understand how musical style is encoded and thus further understand the underpinnings of musical creativity in our own brains. Furthermore, I think my project shows the possibilities of processing audio using deep CNNs.

Encoding of Input Data

The novel part of my implementation lies in my encoding of the music. I chose to use the short time fourier transform representation of the musical signal (see https://ccrma.stanford.edu/~jos/sasp/Mathematical_Definition_STFT.html for details), which is the first step in generating the spectrogram of a song, to represent a song as a function of two dimensions: frequency and time. Thus you can visualize the song's frequency content in a picture. I implemented the forward and backward STFT using a rectangular window (no shaping), with no overlap. Additionally, I used the red channel for the real part of the STFT and the green for the imaginary, such that I retained as much of the audio data as possible for reconstruction. Finally, it is important to note that I did not apply a logarithmic function to the STFT images, as is common practice while generating a spectrogram for audio processing; this is because a logarithmic function is concave, so it makes contrasts in the image less severe and also amplifies error when transforming back to audio using an exponential function. Instead I chose to keep the data in its linear form, and as a result my image encodings of the music are mostly dark, with only the tones being sung or played highlighted.

Methods

I then used the audio data, transformed into a visual representation via STFT, as inputs to the artistic style transfer algorithm.

As for the actual deep learning algorithm, I used exactly the same algorithm as I implemented in our homework for neural art style transfer. The algorithm uses a pre-trained VGG19 CNN trained for image recognition on ImageNet, then runs both the style and content images through the network and retrieves the activations at several of the convolutional layers throughout the model; in my case this was layers conv1_1, conv2_1, conv3_1, conv4_1, conv4_2, and conv5_1. Conv4_2 was used for the content representation while the other five activations were used for the style representation. In accordance with Gatys' method, then, I took the cost function to be a linear combination of the Gram matrix

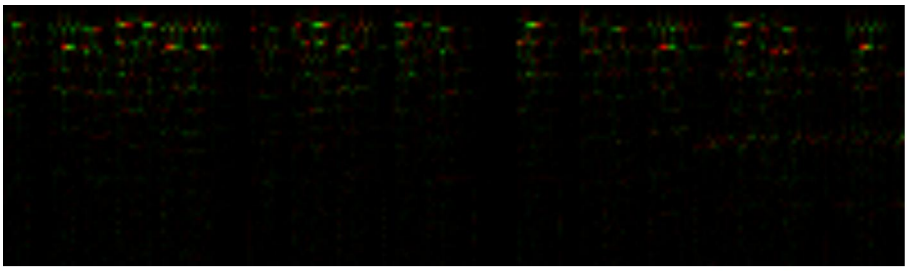
of the difference between activations of each convolutional layer listed above for the style and generated images, unrolled into a vector, and summed over all the convolutional layers with equal weighting among all five; and the Euclidean distance between the conv4_2 activation for the content and generated images. Thus by running gradient descent on the generated image input into the network, I could create an image that was close in style to the style image and in content to the content image.

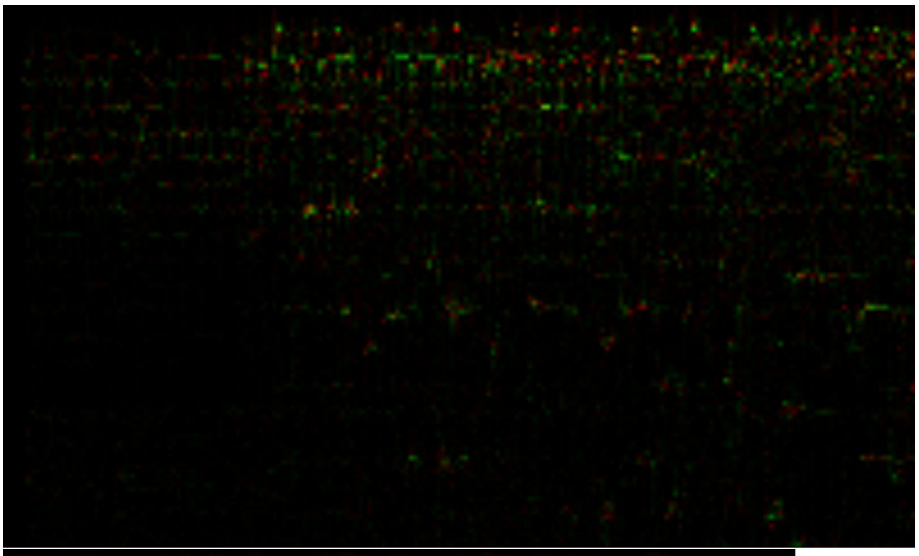
Results and Discussion

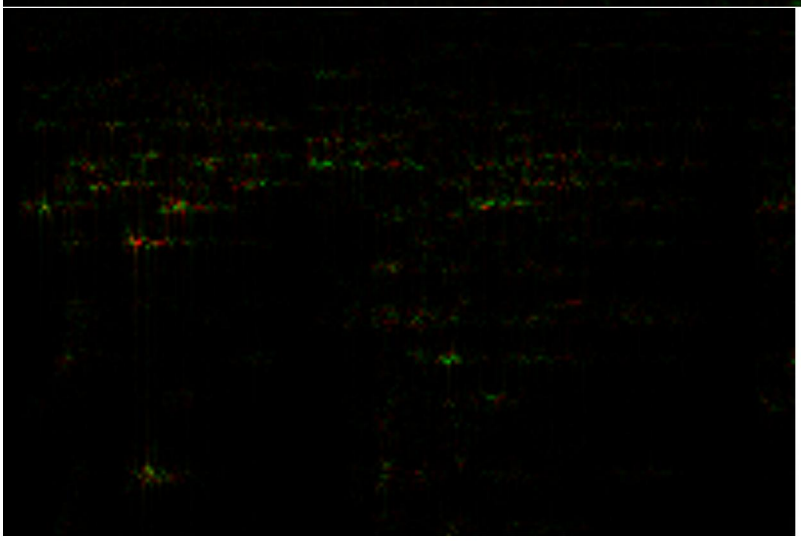
I chose to use a window length of 0.1s for the STFT encoding, since that seemed like a good trade off between frequency resolution and time resolution in the reconstructed songs. I experimented with the various hyperparameters in the deep learning portion of the code but found that I did achieve the best results with equal distribution among the various style layers, and did not experiment much with the reference content layer. Furthermore, I found that it was desirable to have beta in the loss function (the weighting on style loss) be significantly larger than alpha (the weighting on content loss), but did not find that much of a difference in varying the ratio to be larger than 4:1, and with large beta the model had difficulty actually performing the minimization due to unfavorable gradients. Finally, I trained each image for 420 iterations since I saw the loss start to increase again with further iterations.

Below are the STFT images for the songs I used in testing. Note most of the frequency content is in the lower frequencies, towards the top of the image. Time is the x-axis, and each image corresponds to 20s of audio. The images are different heights because I cropped the majority of the dark part out for easier visibility; however, the top of the image is located at DC frequency in each case.

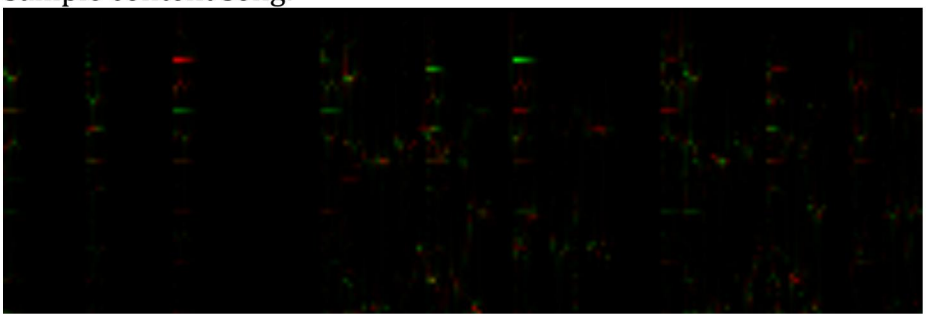
Sample style songs:







Sample content song:



Generated songs (corresponding to style songs above):

Though the generated images might look quite similar, the audio is somewhat different, and picks up on the character of the style song, though not to the extent that I aimed for when beginning the project.
Here are the corresponding audio files (**located in google drive https://drive.google.com/drive/folders/1AFJfFjXjeFNaRuFct6GK38sCgjgV9mDe?usp=sharing**):

Original files (I only used the first 20 seconds)

Style 1: 01 Tripartite Pact.m4a
Style 2: har095-03_i_love_seattle.m4a
Style 3: 2-22 Bach Liebster Jesu, Wir Sind Hier.m4a
Content: 01 Miles Miles.wav

Original files converted via STFT and back to audio

Style 1: Hauff_01_short.wav
Style 2: tacocat_losttime_03_short.wav
Style 3: Bach_2_22_short.wav
Content: YACHT_01_short.wav

Generated

Generated 1: YACHT_as_Hauff_short.wav
Generated 2: YACHT_as_tacocat_short.wav
Generated 3: YACHT_as_Bach_short.wav

Conclusion and Future Work

Given more time to work on this project, I think there are two main areas that could really improve the quality of the output audio: first, using a different window function/overlap to achieve a better reconstruction of the audio after passing through STFT; and second, using a VGG16 CNN trained on a large amount of audio data (say, for a transcription task, trigger word detection, etc) as the reference model, rather than the model trained on images. While there were clearly features of the spectrograms that the image-trained model could pick up on that varied among different songs, a CNN trained on audio data would likely be much better at choosing features important to music, such as harmonies and progression over time.

References

Gatys et al, "A Neural Algorithm of Artistic Style," Sept 2015.

CCRMA at Stanford, "STFT: Overview", https://ccrma.stanford.edu/~jos/sasp/Mathematical_Definition_STFT.html